

第3章

プログラム起動までの処理 / 演算時のCPU状態 /
分岐処理 / 周辺機器への書き込み

64ビットArm仮想ボードで C言語プログラムの挙動を見る

土屋 健

実験の概要

C言語のプログラムがCPU上でどのように実行されているかを、QEMUを使って見てみます(図1)。

●本章でトライすること

ここではQEMU環境と実験用のプログラムを準備し、以下を見ていきます。

- 1) プログラム起動までの処理確認
- 2) メモリの状態確認
- 3) 演算実行時のCPUの状態確認
- 4) 分岐処理が行われる際のCPUの状態確認
- 5) 周辺機器(UART)へのデータ書き込み

●仮想環境QEMU

QEMUでは、多数のCPUや周辺デバイスのエミュレーション環境がサポートされています。今回はその中から、64ビットArmアーキテクチャの仮想ボード・タイプ(virt)を使ってみます。

<https://qemu.readthedocs.io/en/latest/system/arm/virt.html>

virtデバイスは、実際のハードウェアは存在しない仮想的なArmボードとなっています。参考までにQEMUでサポートされているデバイスは以下のコマンドで確認できます。

```
qemu-system-aarch64 -machine help
```

●実験のプログラム

実験のために用意するプログラムは2つです。

- ・C言語の本体プログラム(test.c)
- ・スタートアップ・プログラム(start.S)

実験の本体プログラムはC言語で実装しています。今回はOSのない組み込みボードでの動作を想定しており、標準ライブラリは使用できないため、そこだけは注意が必要です。実験のためのプログラムなので、分岐処理時の挙動が分かりやすいように、意味のない比較処理なども入っています。

```

VM - qemu-system-a
...machine virt -kernel out.img -S -m
X17=0000000000000000 X18=00000000
X20=0000000000000000 X21=00000000
X23=0000000000000000 X24=00000000
X26=0000000000000000 X27=00000000
X29=0000000000000000 X30=00000000
PSTATE=600003c5 -ZC- EL1h FPU
(qemu)
(qemu) info registers
PC=00000000400800cc X00=00000000
X02=0000000000000000 X03=00000000
X05=0000000000000000 X06=00000000
X08=0000000000000000 X09=00000000
X11=0000000000000000 X12=00000000
X14=0000000000000000 X15=00000000
X17=0000000000000000 X18=00000000
X20=0000000000000000 X21=00000000
X23=0000000000000000 X24=00000000
X26=0000000000000000 X27=00000000
X29=0000000000000000 X30=00000000
PSTATE=600003c5 -ZC- EL1h FPU
(qemu) si
unknown command: 'si'
(qemu) info registers
PC=00000000400800d0 X00=00000000
X02=0000000000000000 X03=00000000
X05=0000000000000000 X06=00000000
X08=0000000000000000 X09=00000000
X11=0000000000000000 X12=00000000
X14=0000000000000000 X15=00000000
VM - lldb-ar
15 while (*msg != '\0') {
-> 16 UARTDR = (unsigned int)*msg;
17 msg++;
18 }
19 return;
Target 0: (out.elf) stopped.
[1]ldb> n
Process 1 stopped
* thread #1, stop reason = step over
frame #0: 0x000000004008002c out.elf`printc
14 {
15 while (*msg != '\0') {
-> 16 UARTDR = (unsigned int)*msg;
17 msg++;
-> 17 msg++;
18 }
19 return;
20 }
Target 0: (out.elf) stopped.
[1]ldb> n
Process 1 stopped
* thread #1, stop reason = step over
frame #0: 0x0000000040080038 out.elf`printc
12
13 void print(char *msg)
14 {
-> 15 while (*msg != '\0') {
16 UARTDR = (unsigned int)*msg;
17 msg++;
-> 17 msg++;
18 }
Target 0: (out.elf) stopped.
[1]ldb> n

```

図1 仮想環境QEMUを利用してC言語プログラムの挙動を見える

▶C言語の本体プログラム(リスト1)

本体のmain関数(22～49行目)では、加算演算、条件分岐、メッセージ出力を行う処理を実装しています。具体的には後ほど動作確認します。

24～25行目でメッセージ文字列を定義しています。ここではメッセージそのものは初期化データ・セグメントに配置され、その領域を指すポインタ変数がスタックに確保されます。

31行目は加算演算です。

34行目のif文は後続の条件分岐の実験をするために、ステータス・レジスタの条件フラグをNにするためのものです。これがないとここに到達するまでに条件フラグにZが立った状態となっており、後続の動作確認ができないので入れています。

39行目のif文は実装上は必ず真になる処理でENDラベルに飛びます。

47行目のprint関数で指定文字列をシリアル・コンソールに出力して処理は終了となります。

print関数(13～20行目)では、PL011 UARTデバイスに対して文字列を1文字ずつループして書き出している処理です。これでコンソールに文字が表示されます。